Two-Byte Serial

EGR 445/545

Big Ideas

- The Python+Arduino approach makes our life easier
  - Python is easier than C
    - I will fight anyone who says differently
- We need to transmit numbers from 1000-2000
- Serial is the easiest way to communicate between Python and Arduino
- Serial is built around single byte transmissions
- We will need to break integers into two byes, transmit the two bytes separately, and then reassemble them.
- It really isn’t that hard.

Python+Arduino Servo Control

- We will use serial communication between Python and Arduino
- The Arduino servo module uses periods between pulses to tell the servo the desired position
- Periods can be between 1000-2000 microseconds
- Python will determine the correct period for each servo based on inverse kinematics

Serial Communication Challenge

- Serial is built around transmitting bytes
- What is a byte?
- Why can’t we use one byte per servo to transmit the period?

What is a byte?

- 8 binary digits:

- What range of numbers can be represented by 8 binary digits (i.e. one byte)?

Core Challenge

- Each byte is a number between 0-255 or an ASCII character
- How do we transmit numbers between 1000 and 2000 using as few bytes as possible?

Two-Byte Integers

- Break each integer that you want to transmit into two bytes
  - Most significant byte (msb)
  - Least significant byte (lsb)
- `myint = msb*256 + lsb;//used to reassemble`
  - Find `msb` from integer division:
    - `msb = int(myint/256)`
  - `lsb` is the remainder:
    - `lsb = myint - msb*256`
  - Python has a “remainder” operator:
    - `lsb = myint % 256`

Two-Byte Range

- What range of numbers can be represented using two-byte integers?